#7

Exhibit A
US Serial No.:
10/017,435

EXPRESS MAIL LABEL NO.:

EL830058757US

VOICE APPLICATION DEVELOPMENT METHODOLOGY

Mark Phillips
Jonathan Cook
Pericles Haleftiras
Lizanne Kaiser
Jon Thomas Layton

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to voice applications and, more particularly, to a methodology of using a voice interface platform, having generic software components, and a deployment environment to develop and deploy a specific voice application.

Cross-Reference To Related Applications

This application is a continuation-in-part of United States Application Serial No. 09/855,004, filed May 14, 2001, entitled "Voice Integration Platform," which is a continuation in part of co-pending United States Application Serial No. 09/290,508, filed April 12, 1999, entitled "Distributed Voice User Interface." These co-pending applications are assigned to the present Assignee and are incorporated herein by reference.

Description of the Related Art

A voice user interface (VUI) allows a human user to interact with an intelligent, electronic device (e.g., a computer) by merely "talking" to the device. The electronic device is thus able to receive, and respond to, directions, commands, instructions, or requests issued verbally by the human user. As such, a VUI facilitates the use of the device.

A typical VUI is implemented using various techniques that enable an electronic device to "understand" particular words or phrases spoken by the human

user, and to output or "speak" the same or different words/phrases for prompting, or responding to, the user. The words or phrases understood and/or spoken by a device constitute its "vocabulary."

It is known to develop voice applications for a variety of uses. Voice applications allow users to interact with a user via a variety of communication methods. Voice applications enable companies to provide users with information in an efficient manner, often without the need for any type of operator involvement.

It is known to custom-develop voice applications for specific uses. The custom development of a voice application includes a plurality of phases. These phases include a dialog design phase, a script development phase, a grammar development phase, a prompt development phase, an integration phase, a system test phase and a deployment phase.

The dialog design phase designs how a user will interact with the voice application. The design is represented as a set of dialog flows, which capture scenarios of user interaction. Linguistic knowledge may be used to understand how to prompt a caller to return a predictable response. The more predictable the response, the better the voice application will perform. During the execution of the dialog design phase, key phases of the voice application will be realized including scripts, prompts and grammars.

After the dialog design is completed a script development phase is used to build a dialog interaction. Grammars and prompts are often used to test the scripts and thus the grammars and prompts are often developed in parallel with the scripts. A script developer may use Text-to-speech (TTS) and limited grammars to test the basic functionality of the scripts.

After the script is developed, a grammar is developed during a grammar development phase. In voice applications that do not use DTMF as the primary caller communication mechanism, Automatic Speech Recognition (ASR) is used to recognize what a caller has spoken and to translate that into the action that the caller wishes to take. The recognition rules for performing these functions are captured within an ASR grammar, which maps a set of allowed utterances to a set of appropriate actions or interpretations. Grammars are specific, i.e., different ASR engines may process rules in different ways.

After a grammar is developed then prompts are developed during a prompt development phase. When interacting with a caller, a voice application should speak to the caller. It is known to speak to a caller using TTS technology or using pre-recorded speech. In the latter case, a speech talent records phrases that are required as defined within a dialog design specification. It is usual for these prompts to be recorded in an audio format that is not optimized for the telephony environment. Often processing is used to convert the audio format. This conversion is generally performed by a sound processing engine. Also, audio files are often concatenated to build full phrases used by an application as independently recording every utterance would be inefficient.

After the grammar is developed, then during an integration phase a voice application is integrated to retrieve data from an external resource to be presented to a caller.

After the voice application is integrated, then the voice application is tested and debugged. Often this testing is performed using manual testing techniques. However, some automated tools can be programmed to perform test suites.

Once tested then the voice application is ready to be deployed. Applications are must often be deployed on multiple platforms. For example, parts of an application may be deployed to a script server, a voice gateway, an ASR server, etc. Often when deployed, the application environment is replicated for redundancy or scalability.

What is needed is a platform that allows development and deployment of custom voice applications using generic software components.

SUMMARY OF THE INVENTION

The above and other objects, advantages and capabilities are achieved in one aspect of the invention providing a method that comprises the step of utilizing one or more generic software components to develop a specific voice application. The generic software components are configured to enable development of a specific voice application. The generic software components include a generic dialog asset that is stored in a repository. The method further comprises the step of deploying the

specific voice application in a deployment environment, wherein the deployment environment includes the repository.

The foregoing is a summary and this contains, by necessity, simplifications, generalizations and omissions of detail; consequently, those skilled in the art will appreciate that the summary is illustrative only and is not intended to be in any way limiting. As will also be apparent to one of skill in the art, the operations disclosed herein may be implemented in a number of ways, and such changes and modifications may be made without departing from this invention and its broader aspects. Other aspects, inventive features, and advantages of the present invention, as defined solely by the claims, will become apparent in the non-limiting detailed description set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and it's numerous objects, features and advantages made apparent to those skilled in the art by referencing the accompanying drawings, in which:

FIGURE 1 illustrates a voice application design methodology having a design stage and a deployment stage.

FIGURE 2 illustrates a voice application design methodology wherein the design stage has a plurality of operations.

FIGURE 3 illustrates the additional design stages necessary when using custom software components to develop a voice application.

FIGURE 4 illustrates a design methodology platform according to at least one embodiment of the present invention.

FIGURE 5 illustrates a voice application deployment environment.

FIGURE 5A illustrates a dialog control component architecture.

FIGURE 6 illustrates a repository that is available during the design phase.

FIGURE 7 illustrates at least one embodiment of an architecture for a repository.

FIGURE 8 illustrates at least one embodiment of an architecture for a dialog component.

FIGURE 9 illustrates at least one embodiment of a component load sequence.

FIGURE 10 illustrates at least one embodiment of an architecture for a prompt engine.

FIGURE 11 illustrates at least one embodiment of prompt generation sequence.

FIGURE 12 illustrates at least one embodiment of an architecture for a messaging services layer.

FIGURE 13 illustrates at least one embodiment of a message-send sequence.

FIGURE 14 illustrates at least one embodiment of a message-receive sequence.

FIGURE 15 illustrates at least one embodiment of an architecture for a rules integration layer.

FIGURE 16 illustrates at least one embodiment of an architecture for a detail tracking layer.

FIGURE 17 illustrates a distributed voice user interface system, according to an embodiment of the present invention.

FIGURE 18 illustrates details for a local device, according to an embodiment of the present invention.

FIGURE 19 illustrates details for a remote system, according to an embodiment of the present invention.

FIGURE 20 is a flow diagram of an exemplary method of operation for a local device, according to an embodiment of the present invention.

FIGURE 21 is a flow diagram of an exemplary method 200 of operation for remote system.

DETAILED DESCRIPTION

For a thorough understanding of the subject invention, reference may be had to the following detailed description, including the appended claims, in connection with the above-described drawings.

Turning first to the nomenclature of the specification, the detailed description which follows is represented largely in terms of processes and symbolic representations of operations performed by conventional computer components, such

as a central processing unit (CPU) or processor associated with a general purpose computer system, memory storage devices for the processor, and connected pixel-oriented display devices. These operations include the manipulation of data bits by the processor and the maintenance of these bits within data structures resident in one or more of the memory storage devices. Such data structures impose a physical organization upon the collection of data bits stored within computer memory and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

For purposes of this discussion, a process, method, routine, or sub-routine is generally considered to be a sequence of computer-executed steps leading to a desired result. These steps generally require manipulations of physical quantities. Usually, although not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for those skilled in the art to refer to these signals as bits, values, elements, symbols, characters, text, terms, numbers, records, objects, files, or the like. It should be kept in mind, however, that these and some other terms should be associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as adding, comparing, moving, or the like, which are often associated with manual operations performed by a human operator. It must be understood that no involvement of the human operator may be necessary, or even desirable, in the present invention. The operations described herein are machine operations performed in conjunction with the human operator or user that interacts with the computer or computers.

In addition, it should be understood that the programs, processes, methods, and the like, described herein are but an exemplary implementation of the present invention and are not related, or limited, to any particular computer, apparatus, or computer language. Rather, various types of general purpose computing machines or devices may be used with programs constructed in accordance with the teachings

described herein. Similarly, it may prove advantageous to construct a specialized apparatus to perform the method steps described herein by way of dedicated computer systems with hard-wired logic or programs stored in non-volatile memory, such as read-only memory (ROM).

The methodology described herein is designed to take advantage of a development platform, sometimes referred to herein as a "design methodology platform," and deployment environment in order to develop, deploy, and manage specific voice applications. The methodology facilitates faster, easier application development and deployment than is realized with traditional custom-generation techniques. The methodology utilizes a development platform to develop a specific voice application. The methodology further utilizes a deployment environment to deploy the specific voice application. One of skill in the art will note that certain features are available to a developer during the development stage (via the development platform) and are also available in the deployment environment during the deployment stage. Such overlap of functionality during different phases of the methodology does not necessarily imply duplicate features but instead may be seen as providing access to the same features during different stages of the methodology.

Voice Application Design Methodology Overview

Referring to Figure 1, a voice application is developed in accordance with a design methodology and, in the preferred embodiment, is developed using a design methodology platform 2 (FIGURE 4). The design methodology includes a design phase 1102 and a deployment phase 1104. That is, once a specific voice application is developed and tested in the design phase 1102, the voice application is deployed at deployment step 1104. Once the specific voice application is deployed at deployment step 1104, then the specific voice application is maintained in an iterative fashion as illustrated in FIGURE 1. The regular ongoing maintenance improves voice recognition accuracy and usability. The maintenance includes analyzing data from the specific voice application and making adjustments to the dialog flow and grammars to improve the user's overall user experience with the specific voice application.

Referring to FIGURE 2, the design phase 1102 of the methodology includes the utilization of a plurality of generic software components in order to develop a specific voice application. Such generic software components comprise a design methodology

platform 2 (FIGURE 4). More specifically, the design phase 1102 of the voice application design methodology includes utilization of various components of the platform 2. The design phase 1102 of the methodology includes a dialog design phase 1202, a voice coding phase 1204, a personalization phase 1206, a custom prompt development phase 1208, a custom grammar development phase 1210, a standard prompt phase 1212, a standard grammar phase 1214, a system test phase 1216.

The dialog design phase 1202 involves designing how the user is expected to interact with the voice application that will be deployed. For example, the dialog design phase 1202 may include the definition of prompts that are designed to elicit a predictable response from the user. Dialog flows may be generated, the dialog flows capturing and defining the anticipated scenarios of user interaction with the voice application that is to be deployed. These dialog flows are developed into scripts. In this manner, during the dialog design phase 1202, a voice application user defines the dialog for the specific voice application under development

After the dialog has been designed in the dialog design phase 1202, software application code is generated by the voice application developer during the voice coding phase 1204. During this phase, the voice application developer may utilize a software editing tool, such as Dreamweaver™ by Macromedia, Inc., to generate the basic voice application software.

Briefly also referring to FIGURE 5, when coding the application during the voice coding phase 1204, the application developer may choose to invoke certain pre-designed generic dialog components 540, discussed in further detail below. The dialog components 540 include getDigits, transfer, goodbye, getInfo, getNumber, and getTime components. If he desires to utilize one or more dialog components 540, the application developer develops software code that will cause the specific application 27 to invoke the desired generic dialog components 540 from the deployment environment 500. A similar procedure is followed for invocation of certain pre-designed generic prompt components 1108 (FIGURE 11) provided by the deployment environment 500, including those that provide for combination of individual prompts, pauses of silence, randomization of similar prompts, and prompts based on dynamic data. For further discussion of the prompt components 1108, please see the discussion of FIGURES 10 and 11, below.

Returning to FIGURE 2, it is seen that in the next phase 1206, personality rules are defined in order to impart the desired personality features to the voice application under development. During this phase 1206, a rules definition tool may be utilized by the voice application developer in order to provide for personalization of grammar, prompts, and voice-code flow.

In the prompt development phases 1208, 1212, the developer of the voice application to generates utterances, or prompts, that will be spoken by the voice application to the user. These prompts can be generated using Text-to-speech (TTS) technology and can also be generated using pre-recorded speech. For a voice application that is generated using predesigned generic prompts, phase 1212 is executed. In contrast, custom prompts can be generated in the speech prompts recording phase 1208. In phase 1212, generic pre-generated prompts are used. In phase 1208, a speech talent records prompts as defined by the scripts generated in the dialog design phase 1202.

The design methodology also includes grammar development phases 1210, 1214, wherein the developer of the voice application generates grammars. Grammars define which user utterances the voice application recognizes, and provides that the user utterances be translated into a determination of what action the user would like to invoke in the voice application. A grammar is a set of recognition rules. A grammar maps a set of allowed utterances to a set of appropriate actions (or interpretations). One or more grammars for a voice application may be generated using the standard grammar component 1214. Similarly, one or more grammars for a voice application may be generated using the custom grammar development component 1210.

One skilled in the art will recognize that the phases 1202, 1204, 1206, 1208, 1210, 1212, 1214, 1216 are not necessarily sequential and need necessarily be performed in the order described. For instance, scripts are generally developed after the dialog design phase 1202 is completed. Since grammars and prompts are ordinarily required to test the scripts, they are normally developed in parallel, with the script developer using text-to-speech and limited grammars to test the basic functionality of the scripts.

The final phase of the application design phase 1102 is the system test phase 1216. During this phase, the voice application generated in the voice coding phase

1204 is integrated with data retrieved from an external resource to be presented to the user. The voice application generated in the voice coding phase 1204 may be integrated with the generic standard prompts and grammars identified in phases 1212, and 1214. Alternatively, the voice application generated in the voice coding phase 1204 is integrated with the custom-generated speech prompts and grammar developed in phases 1208 and 1210. Alternatively, the voice application may be integrated with a combination of generic and custom components during the system test phase. During the integration phase, a specific voice application 27 (Figure 4) is thus generated.

The integrated specific voice application 27 (Figure 4) is tested during the system test phase 1216 to confirm that it operates as expected.

FIGURES 2 and 4 illustrate that a design methodology platform 2, described in further detail in the designated section below, may be utilized to facilitate faster and easier performance of the phases of the design methodology described herein. As is described below, the design methodology platform 2 provides generic software components that may be utilized by the voice applications developer during the design phase 1102. For instance, the design methodology platform provides a set of generic, reusable voice application components for commonly used functions that may be used to facilitate easier and faster execution of the dialog design phase 1102.

FIGURES 2 and 3 illustrates that generic components 310 are available for the voice application developer's use during the standard prompts 1212 and standard grammar 1214 phases of the design phase 1102. In contrast, custom prompts and grammars are created in phases 1208 and 1210 following the method illustrated in operations 312 through 316 in FIGURE 3. The custom components are coded by the voice application designer in operation 312. The custom-coded components are then tested in a standalone environment in operation 314. If tested successfully, the custom-coded components are then integrated with the voice application. In operation 318, the completed application is tested, regardless of whether the application used standard components or custom-coded components.

Deployment Environment Overview

FIGURE 5 illustrates a deployment environment 500. The deployment environment 500 provides various architectural elements 4, 512 that provide an

environment for deployment of a specific voice application 27 (FIGURE 4) generated as a result of the methodology described above in connection with Figures 1, 2, and 3. The deployment environment 500 includes a voice gateway 4. The voice gateway 4 handles interactions with the telephony channel responding to events generated by the user of the specific voice application 27. Because the user communicates with the voice application 27 deployed in the deployment environment 500 via the public switched telephone network ("PSTN") 510, the user is referred to herein as a "caller."

The gateway 4 includes a voice interpreter 520. The interpreter requests information from a specific voice application 27 running on an application server 512. In at least one embodiment, the interpreter 520 requests pages from the application server 512 using the hypertext transfer protocol ("HTTP") much in the manner that a web browser requests pages from an internet site. In order to provide the desired input/output functionality to the caller, the voice gateway 4 includes a telephony interface 523 and supports a number of different media services including automatic speech recognition (ASR) 522 and text-to-speech (TTS) services 521. In a preferred embodiment, the voice gateway 4 supports ASR services provided by SpeechWorks International, Inc., of Boston, Massachusetts and Nuance Corporation of Menlo Park, CA. In a preferred embodiment, the voice gateway 4 supports TTS services provided by SpeechWorks and Fonix Corporation of Salt Lake City, Utah.

FIGURE 5 illustrates that the deployment environment 500 also includes a dialog control module 525. In a preferred embodiment, the dialog control module 525 is a software module residing on the application server 512. The dialog control module 525 controls application requests as well as requests for components. The dialog control module 525 maps incoming data requests from the voice gateway 4 to the appropriate logic and data in order to provide the appropriate information to the user, based on that incoming request. In at least one embodiment, the incoming request is associated with a Uniform Resource Identifier (URI) that identifies an object in an internet-based environment.

FIGURE 5A illustrates at least one embodiment of an architecture for the dialog control component 525. HTTP requests are received by a dialog controller 526 and routed to an appropriate sub-controller 15, 527, 528, 529 based on the URI of the request. This mapping is configured using a configuration file 530 that contains the URIs that are expected by the application 27 (FIGURE 5). FIGURE 5A illustrates that,

in addition to the generic sub-controllers 527, 528, 529 provided in the deployment environment 500, application developers can custom-build their own controller 529 to perform specific tasks should that be required.

The dialog control component 525 illustrated in FIGURES 5 and 5A perform control functions as follows. The dialog controller 526 manages startup of the voice application 27 when the call is first received. This is done by configuring the one or more available applications 27 within the deployment environment 500 with their associated startup parameters. When a call is received from the voice gateway 4 the dialog controller 526 initializes an application context and routes the request to an appropriate application script (not shown).

The dialog controller 526 also performs request routing. The dialog controller 526 receives requests from the voice gateway 4 for application and component resources. Each request is then translated into an action, which is then executed. Determination of the action to be executed is based on the configuration file 530. The configuration file 530 maps inbound URI information to a specific class for processing. In at least one embodiment, the specific class is a Java action class. When the dialog controller 526 receives the request it may also receive request parameters. If request parameters are received, the dialog controller 526 forwards them to the appropriate action class. In at least one embodiment, the dialog controller 526 maps the request parameters into a Java container class, which is then routed with the request to the appropriate action class.

The dialog controller 526 can be programmed, in at least one embodiment, to handle the switching between different domains (not shown) within the specific application 27. Domain level requests are passed to the dialog controller 526 which then executes a set of domain control rules to determine which voice application domain should be loaded next. These domain control rules are provided as part of the application configuration, and are implemented using a dynamic rule set.

FIGURES 4, 5, and 6 illustrate that various assets of the design methodology platform 2 may be stored in a voice application repository 600. The voice application repository 600 provides a consistent storage location for dialog assets that can be managed by the voice application developer during the dialog design phase 1102, and that can be deployed to multiple platforms quickly and easily. For this reason, the

voice application repository is designed to organize common dialog assets. Some of these common dialog assets include scripts, prompts, audio files, grammars, and prompt pools. The scripts are re-usable generic scripts that can be used to generate the specific voice application 27 during the voice coding phase 204 of the design methodology. Prompts are scripted, reusable prompts that can be used at any point within the dialog flow of the specific application 27. The audio files are voice files, such as .WAV files, that are used to interact with the caller. Grammars are ASR grammars that are used as the rule base for recognizing utterances spoken by the user.

FIGURE 7 illustrates that the repository 600 includes a remote repository interface 710. The remote repository interface 710 provides an interface by which external applications may access information stored within the repository 600. In at least one embodiment, the remote repository interface 710 is implemented as an Enterprise Java Bean (EJB). The remote repository interface 710 is responsible for lookup up assets based upon a unique key. The unique key consists of a collection and an asset name. The collection is defined as the directory that the file resides in within the repository 600. The files reside in a file system 740. The directory is akin to directories in a standard file system. The asset name represents the unique asset name of the asset within the collection.

The repository 600 groups all of the stored components by application and by language. The repository 600 thus facilitates management of multiple voice applications 27 and multiple languages (such as English, French, etc.) dynamically. In addition, the repository 600 allows meta-data to be stored with any dialog asset through the use of a file representation specific to the dialog asset type. In a preferred embodiment, the meta-data is stored in the form of an XML descriptor.

FIGURE 7 illustrates that the Repository structure 720 includes a repository asset factory 722, asset objects 724, a binary file interface 726, a database 728, and a cache 730. The database 728 facilitates relatively speedy searching and retrieval of asset meta-data. The asset objects 724 are used by the voice application developer during the design phase 1102 (FIGURE 1) and are also used during deployment 1104 (FIGURE 1). Asset objects 724 include scripts, prompts; components, audio objects, grammars, and prompt pools. A binary file interface 726 facilitates storage and retrieval of binary files that are stored in the repository, such as audio files. The repository 600 supports streaming of such binary audio files to the external

application from a remote storage location. This remote-streaming capability allows the repository files to reside on a central repository if so required by an application.

The cache 730 provides for improved retrieval latency for items that have been previously retrieved from the file system 740. In at least one embodiment, external applications may add and update files directly within the file system 740.

FIGURE 5 illustrates that the deployment environment 500 further includes one or more dialog components 540, discussed briefly above in connection with FIGURE 2. A plurality of dialog components 540a - 540n may include the following:

- GetDigits

  Gets a specified number of digits from the user (e.g., "one two three four"). Can be parameterized to recognize a string length between 1-10 digits. Supports optional DTMF recognition. Includes optional confirmation state.

- Transfer

  Transfers call to another number. Will alert caller if number is busy or there is no answer. Can be parameterized such that if the call connects, at the end of the call the system either returns to the voice application or disconnects.

- Goodbye

  Recognizes "goodbye" and ends phone session. Provides the user with the opportunity to cancel a "goodbye". This is critical in cases where user input could be misinterpreted as goodbye.

- GetInfo

  Gets some information, the content of the information determined by the developer. Since the content of the dialog is not predictable, the product release will contain some example prompts and grammar that the developer can use as a model. Could be adapted to get a single piece of information (e.g., "How many would you like to order?") or to select from a menu of items (e.g., "Would you like National News, World News, or Sports News?"). Includes optional confirmation state.

- GetNumber

  Gets a natural number from the user (e.g., "one-thousand two-hundred thirty-four"). Range from 0 to 1,000,000. Supports optional DTMF recognition. Includes optional confirmation state.

- GetTime

  Gets time from the user. Recognizes the hour, minutes, and am/pm. Includes optional confirmation state.

- GetCurrency

  Gets currency amount from the user in U.S. dollars and cents. Supports optional DTMF recognition. Includes optional confirmation state.

- YesNoConfirmation

  Gets yes or no reply from the user in response to a question. Supports optional DTMF recognition.

- Login

  Accepts a numeric user ID number and optional numeric PIN and passes the information to a backend validation mechanism. Supports optional DTMF recognition. Includes optional confirmation states.

- GetPhoneNumber

  Gets a 10-digit phone number. Supports optional DTMF recognition. Includes optional confirmation states.

- BrowseList

  Allows the user to navigate through a list of items (e.g., "Get the next one." "Go to the first one." "Previous one." "Last one, please."). Allows the user to select an item from that list and perform some action. Optionally reads the items in the list sequentially without user initiation (auto navigation).

- GetDate

  Gets date from the user, including day, month and year. Includes optional confirmation state.

- GetCreditCard

  Gets credit card number. Length of credit card number can be parameterized. After getting credit card number, there is an optional confirmation state for the number.

- GetZip

  Gets a United States zip code from the user.

FIGURE 8 illustrates the architecture of each component 540a-540n. In at least one embodiment, each dialog component 540 is a reusable, extensible Java object that includes a component definition file 802, a component script 804, a dialog form object 806 and a component data access layer 810. Each dialog component 540 loads parameters from a component parameter set 808.

The component definition file 802 is a file that is used by the component controller 525 to determine the configuration of the component 540. It is used to validate the parameters that are passed by the application 27 to the component, and to determine which script file is loaded.

The component script 804 contains all the logic that is required to present the dialog component 540. Parameters are passed to the component script 804 from the component controller 252. In at least one embodiment, parameters are passed using a

Java class called a form object 806. In at least one embodiment, the component script 804 may utilize custom tag libraries, provided by the voice application services layer 3 to interact with back-end services.

The dialog form object 806 receives the parameters from the component controller 525 that are passed with the request. It is then mapped into the request scope of the component script 804 so that the component script 804 can use that to configure itself.

The component parameter set 808 is a file that contains information required to instantiate a component 540 to achieve a specific purpose. The component parameter set 808 can be created by the user using a graphical editor, such as Dreamweaver™. By providing a
number of readily available parameter sets 808, the deployment environment 500 allows the component 540 to modify its behavior. Using the mechanism, for example, the getDigits components can become a getZip or getCreditCard number by passing in a different parameter set.

The component data access layer 810 is an optional part of the component 540, dependent on the functionality required, and is the interface to the back-end resources that are required by the component 540. The component access data layer 810 utilizes the voice application services 3 to retrieve information from the various back-end data sources and services.

FIGURE 9 illustrates a component load sequence. Figure 9 illustrates that a component is invoked in sequence 902 when a request is issued by the voice gateway 4. The request is passed to the dialog control 525 with the URI of the component 540 to be loaded. The dialog control 525 performs a look-up in the dialog control configuration file 530. The dialog control 525 then fetches the appropriate component 540 from the component definition file 802 in sequence 906 and loads the appropriate component in sequence 908. FIGURE 9 illustrates that, in the remaining sequences 909-942, the dialog control 525 forwards the HTTP request to the component 540, passing the parameters from the request. The component script 504 then performs the function that is required by the application, including accessing any back-end services that are required.

In addition to dialog components 540 the deployment environment 500 supports the invocation of prompt components 1108 by the voice application designer during phase 1212. These are reusable snippets of dynamic code that produce the prompting that is used to communicate with the caller. Each prompt component 1108 is defined and stored within the voice application repository 600 and can be executed by passing a set of input parameters. Prompts are defined within a file, such as an XML file. FIGURE 10 illustrates that, in at least one embodiment, a runtime prompt engine 1000 loads the prompt file from the repository 600 and executes the prompt. FIGURE 10 illustrates the architecture of the prompt engine 1000 and associated prompt component.

FIGURES 10 and 11 illustrate that application scripts and components can include dynamic content from the prompt engine 1000 using a provided custom tag 1004a. FIGURE 11 illustrates that, in sequences 1110-1158, this custom tag 1004a passes the prompt information to the prompt engine 1000 by requesting the servlet dispatcher to include the content created by a request to the prompt servlet 1006. Once the request is received by the prompt engine 1000 the prompt 1108 is loaded and an interpreter context 1008 is created. During this process the prompt servlet 1006 maps all the input variables (request attributes) to the interpreter context 1008. In addition, the output stream of the servlet 1006 is mapped into the context to allow prompt execution to out put dynamic information to the appropriate stream. In at least one embodiment, prompt 1108 can execute to access repository 600 resources and system variables.

Returning to FIGURE 5, it can be seen that the deployment environment 500 includes a voice application services layer 3. It should be noted that the voice application service layer 3 is also available to a voice application developer during the design phase 1102 (FIGURE 1) as part of the design methodology platform 2. The voice application services layer 3 provides access to underlying communication layers such as a messaging layer 539, a rules integration layer 537, a voice services layer 538, and a call detail tracking layer 541. In at least one embodiment, the voice application services layer 3 is implemented as a set of Java™ classes with tag wrappers. Tag libraries are provided for each of the supported underlying services 537, 538, 539.

For instance, a set of custom tags is provided for the application messaging layer 539. The application messaging layer provides for sending and receiving messages from within a voice application script or component. The tags for the application messaging layer enable the sending of a message directly from a script without requiring execution within a server page. Another set of tags is provided for acting as helper tags to facilitate access of assets stored within the repository 600. The tags load information from the repository 600 using the current application context, and provide for utilization of the repository asset during the script execution.

FIGURE 5 illustrates that the messaging services layer 539 allows voice applications 27 to send message requests to external systems 542a, 542b, 542n. In at least one embodiment, the messaging services layer 539 is a set of classes that provide the capability to create and bind data messages and send them to specific destinations through utilization of a number of various protocols. The messaging services provided by the messaging services layer are accessible via the voice application service layer 3. The voice application service layer 3 provides a set of custom tag libraries that provide for interaction with the messaging services from within the voice application 27 (FIGURE 4). In at least one embodiment, the voice application service layer 3 provides a tag library to support the following message protocols: Java™ messaging service ("JMS") and Java™ XML messaging service ("JAXM").

FIGURE 12 illustrates that the messaging services 1222 provided by the messaging service layer 539 include a message sender service 1228 for sending messages. The messaging services 1222 also include a message listener service 1224 for receiving messages. The messaging services 1222 also include a data binding service 1226. The messaging services layer 539 provides an abstraction layer between the voice application services layer 3 and the individual message service 1222.

FIGURES 4, 5, 12, and 13 illustrate that, when an application 27 wishes to send a message 1300 to an external system 542, it can do so by initiating a connection to the message service 1227 and creating a message of the appropriate type. In sequence 1320 the voice gateway 4 sends the request to the voice application services layer 3. In sequences 1304 and 1306 the voice application services layer 3 determines the appropriate messaging controller 122. The messaging services layer 539 creates/allocates an instantiation of the message service 1222. In sequence 1308, the

voice application services layer 3 requests that a message 1300 of the appropriate type be created. The message service 1222 creates the message 1300 in sequence 1310 and returns control to the voice application services layer 3 in sequence 1312. The voice application services layer 3 then initiates binding of the data that was passed from the application 27. The messaging services layer 539 then coordinate the binding of the data (sequences 1314 and 1316 indicate that multiple pieces of data may be involved in the binding process) to the message 1300. Once the message is completed, in sequence 1318 the messaging services layer 539 facilitates sending the completed message to the message service 1222 for delivery. In sequences 1320 through 1328 the message service 1222 sends the bound message 1300 to its destination 542. If the send operation is an asynchronous operation, optional sequence 1324 may be performed in order to wait for acknowledgement of successful message delivery. In operations 1328 and 1330, control is returned through the messaging services layer 539 and voice application services layer 3 to the voice gateway 4. If an acknowledgement of successful asynchronous delivery was not received during a specified timeout period in sequences 1324 and 1326, the messaging controller 1220 will so notify the application 27 when control is returned in sequences 1328 and 1330.

FIGURES 4, 5, 12, and 14 illustrate that the message-receive sequence illustrated in FIGURE 14 differs from the message-send sequence described above in connection with FIGURE 13. The messaging controller 1220 is configured, in a preferred embodiment, to bring up a plurality of message listeners 1224 when the deployment platform 500 is first initialized. The listeners 1224 remain active and will be forwarded any received message on a pre-selected topic, queue, or connection. When an application 27 wishes to receive a message 1300 from an external system 542, it registers a message listener request with the message service 1222 in operations 1410 through 1422. Such sequences result in instructing the message service 1222 to notify , via the voice application services layer 3, the voice application 27 when a message 1300 is received. When a message is received by a listener 1224 in sequence 1402, the message listener 1224 binds the data in the message to the appropriate message object . In sequence 1404 the message listener 1224 notifies the messaging controller 1220 that a message has been received from an external message

source 542 and provides, in sequence 1408, the message 1300 to the message service 1222 for storage pending delivery.

Delivery of the message is coordinated by the messaging controller 1220. If the message is already expected by the application 27 (i.e., the message is expected in response to a previous send request), the messaging controller 1220 pairs the response with the sent request, using a request id mechanism. The messaging controller 1220 then waits for the application 27 to pick up the message, and passages the contents to the application 27.

In at least one embodiment, delivery of the message is not guaranteed because 1) the caller could disconnect the conversation before the message is delivered or 2) the message could take too long to arrive. If the application 27 does not pick up the message within a specified timeout period, the message and originating request are expired.

In such cases (i.e., when a message is expired) , the messaging controller 1220 performs data-expiration monitoring and cleanup. If the application 27 requests to receive a message and there is no message avaialbe, then the messaging controler 1220 notifies the application 27 to wait for a specified timeout period. If the requested message does not become available during the timeout period, then the request becomes "timed-out." In such case, the messaging controller 1220 provides a negative acknowledgement to the message source 542 to indicate the requested message has not been delivered.

One of skill in the art will recognize that the send and receive sequences illustrated in FIGURES 13 and 14 are based on the assumption that the message action is initiated by the application 27 rather than the external system, and that external-side message initiation could easily be implemented within the methodology and platforms described herein. For example, in some instances the messaging controller 1220 may receive an ad hoc message that is not associated with a request by the application 27. In such case, the messaging controller 1220 waits for an application 27 to request receipt of a message on the topic, queue , or connection associated with the ad hoc message. If the messaging controller 1220 receives an application request for the ad hoc message within a specified global timeout period, the messaging controller 1220

will forward the ad hoc message to the requesting application 27. Otherwise, the ad hoc message is "timed-out."

FIGURE 5 illustrates that another set of custom tags is provided to provide access to a rules integration layer 537. This set of tags, referred to as a personalization tag library, allows rules to be set up in a rules engine associated with the rules integration layer 537. This allows a rules set to be invoked and also allows for actions to be taken in the dialog based on the result of invocation of the rules set. The tags in the personalization tag library also provide access to information being racked by the application. The custom tags allow the application developer to execute a rule set on a number of parameters to decide what actions should be performed by the application. Some appropriate uses for this interface are:

- Play content based on user information - In this case the custom tags provide the capability to have the application decide whether to play a piece of content or not. The information used to make this determination includes, for instance, inbound call information, and information stored within the user profile of the caller.
- Change available options - In this case the rules change the available options in the application by modifying the prompts and grammars that can be used to recognize the required commands.
- Transfer out - In some cases, when a caller logs in, the application requires immediate transfer of the caller to a customer service representative. An example, of when such transfer is necessary occurs when the user has an overdue account.

FIGURE 15 illustrates the architecture for the rules integration layer 537. The rules integration layer 537 includes a remote interface 1502 that facilitates communication with the voice application services layer 3. The rules integration layer includes a set of custom tags that perform specific rule-type functions and includes a rules engine 1504. A rule engine pool manager 1506 that performs initialization functions by polling and sharing available rule engine resources. Remote users assert objects into the context of the rule engine 1504 and then execute one or more rule files 1508. The rules service remote interface 1502 maintains state across method calls, allowing multiple objects to be asserted before rules are invoked.

Once the rules have been executed, the rules integration layer 537 returns all objects that were affected by the execution of the rules. In at least one embodiment, the affected objects are returned as an object array.

FIGURE 5 illustrates that another set of custom tags is provided to provide access to a voice services layer 538. The tags provide support for voice-activated services such as email services and security services. Additional tags provide support for voice-activated access to database information.

FIGURE 5 illustrates that another set of custom tags is provided to provide access to a call detail tracking layer 541. The custom tags, referred to as a tracking tag library, provide for tracking of information bout a caller session while the application is executing. To do this, the call detail tracking layer 541 provides for tracking information falling into a number of distinct categories, including call-based information, caller-based information, and event-based information. Call-based tracking involves tracking information about a particular call, including when the call started, dialed number, etc. Caller-based tracking involves tracking information about caller. This tracking feature is used in conjunction with a user profile in order to provide personalized content to the caller. Event-based tracking involves tracking the events that occur within the call flow. For example, event-based tracking will record that a caller has been transferred out of the specific voice application. As another example, event-based tracking will record that a caller has provided an invalid credential.

Figure 16 illustrates an architecture for the detail tracking layer 541. Each tracking feature provides tracking throughout the duration of a call. A database 1600 such as a relational database, queues tracking/logging requests using a queue mechanism 1602, 1604. The tracking/logging requests are delivered by the queue receiver 1604 and are provided, via a remote interface 1606, to a tracking object 1608 for storage. In at least one embodiment, the tracking object 1608 is implemented as an Enterprise Java Bean (EJB). A data access object 1610 is used to plug-in various different storage mechanisms to the tracking object 1608. The data access object 1610 is generated by a data access object factory 1612.

Design Methodology Platform Overview

Figure 4 illustrates a design methodology platform 2, according to at least one embodiment of the present invention. In general, design methodology platform 2 provides voice application design software. When installed on a computer system that includes at least a processor and a memory, the design methodology platform 2 software provides a means for developing a specific voice user interface that is designed to interact with a data system 6.

The voice integration design software of the design methodology platform 2 includes generic software components. These components are reusable, allowing the designer of a voice interface to utilize pre-written code in developing a specific voice application that is designed to interface with the data system 6 to deliver information (i.e., "stored data") to a user. Using the design methodology platform 2, an interface designer, or team of designers, can develop a specific voice application 27, such as a specific voice user interface, that allows one or more human users 29 to interact--via speech or verbal communication--with one or more data systems 6. That is, data stored on the data system 6 is ordinarily not presented to a user 29 via voice interaction. If the data system 6 is a Web application server, for example, the user 29 requests and receives information via a local device 14 (FIGURE 17) using a standard GUI interface to interact with the Web server. The design methodology platform 2 provides a development platform that enables an application designer to create a specific voice application 27 that interacts with stored data on an existing data system 6. As used herein, the terms "connected," "coupled," or any variant thereof, means any connection or coupling, either direct or indirect, between two or more elements; the coupling or connection can be physical or logical.

FIGURE 4 illustrates that the design methodology platform 2 is used to generate a specific voice application 27. The specific voice application 27 is designed to send data to, and receive data from, a data system 6. The data system 6 is any computer system having a memory that stores data. In at least one embodiment, the data system 6 is a Web site system that includes server hardware, software, and stored data such that the stored data can be delivered to a user via a network such as the Internet. In at least one embodiment, the data system 6 is system of one or more hardware servers, and software associated with an internet Web site. In such embodiment, data system 6 typically contains one or more Web application servers that include the necessary hardware and software to store and serve HTML documents, associated files, and

scripts to one or more local devices 14 (FIGURE 17) when requested. The Web application servers are typically an Intel Pentium-based or RISC based computer systems equipped with one or more processors, memory, input/output interfaces, a network interface, secondary storage devices, and a user interface. The stored data includes "pages" written in hypertext markup language (HTML) and may also include attribute and historical data concerning a specific user.

In at least one other embodiment, the data system 6 is a system that supports a customer call center application. Such system includes a memory that stores data associated with one or more customers, and software and hardware that permit access to such stored data. In a traditional customer call center, the stored customer data is manipulated, edited, and retrieved by human operators at computer terminals that have computerized access to the data. In such case, the data may be delivered to the human operators via a mechanism other than the Internet, such as an internal network system. In at least one other embodiment, the data system 6 is an automated banking system. The foregoing specific examples of a data system 6 are for informational purposes only, and should not be taken to be limiting.

FIGURE 4 illustrates that the design methodology platform 2 includes a voice gateway 4. The voice gateway 4, in at least one embodiment, incorporates at least some of the functionality of a distributed voice user interfacedescribed below. The voice gateway 4 allows the user of a local device 14 (Figure 2) to interact with the device 14 by talking to the device 14.

FIGURE 4 illustrates that the voice gateway 4 is designed to work in conjunction with a set of service layers 3, 5, 7, 9, 11, 13. As used herein, the term "service layer" refers to a set of one or more software components that are logically grouped together based on shared attributes. The service layers that interact with the voice gateway 4 include the Applications service layer 3, the Personalized Dialogs service layer 5, and the Infrastructure service layer 7. The design methodology platform 2 also includes a Personalization service layer 9, a Content Management service layer 11, and an Integration layer 13. The latter three service layers 9, 11, 13 is each capable of facilitating interaction between the data system 6 and any of the remaining three service layers 3, 5, and 7. A tools service layer 8 is designed to work in conjunction with the voice gateway 4 and each of the service layers 3, 5, 7, 9, 11, 13. The tools service layer 8 set is a set of software programming tools that allows a

voice application developer, for instance, to monitor, test and debug voice application software code that he develops using the design methodology platform 2.

Using the components of the design methodology platform 2 as a development platform, a voice application designer can develop a specific voice user interface 27 that is designed to integrate with a specific existing data system 6. In at least one embodiment, the existing data system 6 is the set of hardware, software, and data that constitute a Web site. Many other types of data systems are contemplated, including automated banking systems, customer service call centers, and the like.

Applications Service Layer

The applications service layer 3 includes components that add certain functional capabilities to the voice interface developed using the design methodology platform 2. For instance one of the components of the Applications service layer 3 is an email component 23. The email component 23 contains software, such as text-to-speech, speech-to-text, and directory management software, that provides the user the ability to receive and send email messages in voice format. Another component of the Applications service layer 3 is a notification component 25. The notification component 25 provides for handing off information from the voice user interface 27 to the local device 14 (FIGURE 17) of a live operator when a user opts to transfer from an automated voice application to live support.

Personalized Dialogs Layer

The Personalized Dialogs service layer 5 is a group of one or more software components that allow a voice applications developer to incorporate natural language concepts into his product in order to present a more human-like and conversational specific voice user interface . The software components of the Personalized Dialogs service layer 5 implement rules for presenting voice information to a user in order to emulate human dialog. Each of the software components may include various constituents necessary for dialog emulation, such as voice XML scripts, .WAV files and audio files that make up the dialog presented to the user, recognition grammars that are loaded into speech recognition components, and software code for manipulating the constituents as needed. For example, the Personalized Dialogs service layer 5 includes an error-trapping component 17. The error trapping component 17 is software logic that provides that prompts are not repeated when an

error occurs with user voice input. The error trapping component 17 includes code that might provide, upon an error condition, a prompt to the user that says, "I didn't quite get that." If the error condition is not corrected, instead of repeating the prompt, the error trapping component might then provide a prompt to the user that says, "Could you please repeat your selection?" If the error condition is still not corrected, the error trapping component 17 might then provide a prompt that says, "Well, I'm really not understanding you." By providing a series of distinct error-handling prompts rather than repeating the same prompt, a more conversational dialog is carried on with the user than is provided by other voice interface systems.

As another example, the Personalized Dialogs service layer 5 includes a list browse component 19. The list browse component 19 provides for presentation of a list of items to a user. The list browser component implements certain rules when presenting a list of information to a user such that the presentation emulates human verbal discourse.

Using the components of the Personalized Dialogs service layer 5, an application designer can design a voice user interface 27 that presents data to the user from an existing data system 6, presenting the information in a verbal format that is personalized to the particular user. For instance, the voice user interface 27 can be designed to obtain attribute information about the user. This information could come directly from user, in response to prompts, or from another source such as a cookie stored on the user's local device 14 (FIGURE 17). The voice user interface 27 can also be designed to track historical information among multiple sessions with a user, and even to track historical information during a single user session. Using this attribute and historical data, the components of the Personalized Dialogs service layer 5 provide for personalized interaction with the user. For an example that uses attribute data, the voice user interface programmed by the application designer (using the design methodology platform) speaks the user's name when interacting with the user. Similarly, if the user attribute data shows that the user lives in a certain U.S. city, the voice user interface can deliver local weather information to the user. For an example using historical data across more than one session, consider a voice user interface between a user and a data system 6 that provides banking services and data. If the voice user interface 27 tracks historical information that indicates that a user, for 10 out of 11 previous sessions (whether conducting the session using a voice interface or

another interface such as a GUI), requested a checking account balance upon initiating the session, then the Personalized Dialogs service layer 5 provides for offering the checking account balance to the user at the beginning of the session, without requiring that the user first request the data.

The Personalized Dialogs service layer 5 also provides for tracking other historical data and using that data to personalize dialogs with the user. For instance, the service layer 5 can be utilized by the application programmer to provide for tracking user preference data regarding advertisements presented to the user during a session. For instance, in at least one embodiment the design methodology platform 2 provides for presenting voice advertisements to the user. The Personalized Dialogs service layer 2 keeps track of user action regarding the advertisements. For instance, a voice add might say, "Good Morning, Joe, welcome to Global Bank's online service voice system. Would you like to hear about our new money market checking account?" The Personalized Dialogs service layer 5 provides a component that ensures that the format of the ad is rotated so that the wording is different during different sessions. For instance, during a different session the ad might say, "Have you heard about our new money market checking account?" The Personalized Dialog service layer contains a component that provides for tracking how many times a user has heard the advertisement and tracks the user's historical responses to the advertisement. To track the effectiveness of the add, the Personalized Dialogs service layer 5 keeps track of how many users opt to hear more information about the advertised feature. By tracking user responses to various adds, user preference information is obtained. This historical user preference information is forwarded to the data system 6. Likewise, the Personalized Dialogs service layer 5 has access to historical and attribute data concerning a user that has been stored on the data system 6. This data may come from any of several points of interaction, or "touchpoints", between the user and the data system 6, including telephone access to a manned call center, voice or non-voice interaction with the data system 6 from a local device such as a personal computer or wireless device, and voice or non-voice telephone communications. This historical user preference information is also maintained for use by the Personalized Dialogs service layer 5. The historical user preference information, along with preference information from the data system 6 that has been obtained during the user's non-voice interaction with the data system 6, is used to

provide personalized dialogs to the user and to target specific preference-responsive information to the user.

The Personalized Dialogs service layer 5 also includes a scheduling component 21 that provides for scenario-driven personalization. Scenario-driven personalization provides additional interaction with the user even after a voice session has been completed, depending on the types of actions taken by the user during the session. For instance, the scheduling component 21 provides an automated process for forwarding printed material to a user if requested by the user during a session. In addition, in certain specified situations the scheduling component 21 provides a notification (i.e., to a customer representative in a call center) to perform a follow-up call within a specified time period after the initial voice session.

FIGURE 4 illustrates that various components of the Personalized Dialogs service layer 5 may be stored in a voice application repository 600. One class of assets that are stored in, and managed by, the voice application repository are the random prompt pools provided by the random prompt pool component 17 of the personalized dialogs service layer 5. These random prompt pools are collections of prompts that are randomized to allow for the dialog of the specific application 27 to sound more natural and human-like.

Infrastructure Service Layer

The Infrastructure service layer 7 is a group of one or more software components that are necessary for all specific voice user interfaces 27 developed using the design methodology platform 2. For instance, the Infrastructure service layer 7 includes a domain controller software component 15. The domain controller software component 15 manages and controls the organization and storage of information into logically distinct storage categories referred to herein as "domains". For instance, "electronic mail", "sports scores" and "news" "stock quotes" are examples of four different domains. The domain controller software component 15 provides for storage and retrieval of voice data in the appropriate domain. In some instances, a piece of voice data may be relevant to more than one domain. Accordingly, the domain controller software component 15 provides for storage of the voice data in each of the appropriate domains. The domain controller also traverses the stored domain data to retrieve user-specified data of interest.

Personalization Service Layer

The personalization service layer 9 contains software modules that facilitate interaction of the specific voice user interface 27 developed using the design methodology platform 2 with personalization data in the data system 6. For instance, the data system 6 may include code for a personalization rules engine. The personalization rules engine on the data system 6 can also be referred to as an inferencing engine. The inferencing engine is software that accesses and processes data stored on the data system 6. For example, the inferencing engine in a data system that conducts e-commerce may track the types of purchases that a particular user has made over time. Based on this information, the inferencing engine predicts other products or services that might be of interest to the particular user. In this manner, the data system 6 generates a "recommended items" list for a particular user. The Personalization service layer 9 provides a software module that facilitates presentation of the "recommended items" to the user in voice format.

Content Management Service Layer

The Content Management service layer 11 contains one or more software modules that facilitate interaction of the specific voice user interface 27 developed using the design methodology platform 2 with content management software on the data system 6. For instance, a data system 6 that manages a large amount of data may include content management software that classifies each file of data by associating a meta tag descriptor with the file. This meta tag descriptor helps classify and identify the contents of the data file. The Content Management service layer 11 provides a software module that facilitates access by the specific voice user interface 27 developed using the design methodology platform 2 to the content management functionality, including meta tag data, of the data system 6.

The Content Management service layer 11 also contains one or more software components that provide for enhanced management of audio content. For instance, some audio files are streamed from a service to the data system in broad categories. An example of this is the streaming of news and sports headlines to the data system 6 from the Independent Television News network. A content management software component parses the stream of audio content to define constituent portions of the stream. The content management software module then associates each defined

constituent portion with a particular domain. For instance, a sports feed can be parsed into college sports and professional sports items that are then associated with the appropriate domain. For smaller granularity, the college sports items are further parsed and associated with football, baseball, basketball, and soccer domains. In this manner the content management software component provides smaller granularity on content than is provided as a streamed audio feed. One skilled in the art will understand that various types of audio data can be received by a data system 6, including voicemail, weather information, stock quotes, and email messages that have been converted to speech. Therefore, the example concerning sports and news headlines audio feed should not be taken to be limiting.

In at least one embodiment, a content management software component facilitates generation of meta tag data for information received from an audio feed, such as the ITN feed described above. The software component provides for converting the parsed audio files to text. Then, the text files are associated with meta data via interaction with the content management software on the data system 6.

In at least one embodiment, a content management software component provides templates for the creation of dialogs in a specific voice user interface 27. This feature speeds the creation of dialogs and provides a pre-tested environment for dialog creation that ensures that related components, such as recognition grammars and .wav files, are integrated properly.

Integration Service Layer

The Integration service layer 13 is an input/output layer that contains software components for allowing the specific voice user interface 27 and the data system 6 to exchange and share data.

Voice Gateway (Distributed VUI System)

FIGURE 17 illustrates a distributed VUI system 10. Distributed VUI system 10 includes a remote system 12 which may communicate with a number of local devices 14 (separately designated with reference numerals 14a, 14b, 14c, 14d, 14e, 14f, 14g, 14h, and 14i) to implement one or more distributed VUIs. In one embodiment, a "distributed VUI" comprises a voice user interface that may control the functioning of a respective local device 14 through the services and capabilities of remote system 12.

That is, remote system 12 cooperates with each local device 14 to deliver a separate, sophisticated VUI capable of responding to a user and controlling that local device 14. In this way, the sophisticated VUIs provided at local devices 14 by distributed VUI system 10 facilitate the use of the local devices 14. In another embodiment, the distributed VUI enables control of another apparatus or system (e.g., a database or a website), in which case, the local device 14 serves as a "medium."

Each such VUI of system 10 may be "distributed" in the sense that speech recognition and speech output software and/or hardware can be implemented in remote system 12 and the corresponding functionality distributed to the respective local device 14. Some speech recognition/output software or hardware can be implemented in each of local devices 14 as well.

When implementing distributed VUI system 10 described herein, a number of factors may be considered in dividing the speech recognition/output functionality between local devices 14 and remote system 12. These factors may include, for example, the amount of processing and memory capability available at each of local devices 14 and remote system 12; the bandwidth of the link between each local device 14 and remote system 12; the kinds of commands, instructions, directions, or requests expected from a user, and the respective, expected frequency of each; the expected amount of use of a local device 14 by a given user; the desired cost for implementing each local device 14; etc. In one embodiment, each local device 14 may be customized to address the specific needs of a particular user, thus providing a technical advantage.

Local Devices

Each local device 14 can be an electronic device with a processor having a limited amount of processing or computing power. For example, a local device 14 can be a relatively small, portable, inexpensive, and/or low power-consuming "smart device," such as a personal digital assistant (PDA), a wireless remote control (e.g., for a television set or stereo system), a smart telephone (such as a cellular phone or a stationary phone with a screen), or smart jewelry (e.g., an electronic watch). A local device 14 may also comprise or be incorporated into a larger device or system, such as a television set, a television set top box (e.g., a cable receiver, a satellite receiver, or a video game station), a video cassette recorder, a video disc player, a radio, a

stereo system, an automobile dashboard component, a microwave oven, a refrigerator, a household security system, a climate control system (for heating and cooling), or the like.

In one embodiment, a local device 14 uses elementary techniques (e.g., the push of a button) to detect the onset of speech. Local device 14 then performs preliminary processing on the speech waveform. For example, local device 14 may transform speech into a series of feature vectors or frequency domain parameters (which differ from the digitized or compressed speech used in vocoders or cellular phones). Specifically, from the speech waveform, the local device 14 may extract various feature parameters, such as, for example, cepstral coefficients, Fourier coefficients, linear predictive coding (LPC) coefficients, or other spectral parameters in the time or frequency domain. These spectral parameters (also referred to as features in automatic speech recognition systems), which would normally be extracted in the first stage of a speech recognition system, are transmitted to remote system 12 for processing therein. Speech recognition and/or speech output hardware/ software at remote system 12 (in communication with the local device 14) then provides a sophisticated VUI through which a user can input commands, instructions, or directions into, and/or retrieve information or obtain responses from, the local device 14.

In another embodiment, in addition to performing preliminary signal processing (including feature parameter extraction), at least a portion of local devices 14 may each be provided with its own resident VUI. This resident VUI allows the respective local device 14 to understand and speak to a user, at least on an elementary level, without remote system 12. To accomplish this, each such resident VUI may include, or be coupled to, suitable input/output devices (e.g., microphone and speaker) for receiving and outputting audible speech. Furthermore, each resident VUI may include hardware and/or software for implementing speech recognition (e.g., automatic speech recognition (ASR) software) and speech output (e.g., recorded or generated speech output software). An exemplary embodiment for a resident VUI of a local device 14 is described below in more detail.

A local device 14 with a resident VUI may be, for example, a remote control for a television set. A user may issue a command to the local device 14 by stating "Channel four" or "Volume up," to which the local device 14 responds by changing

the channel on the television set to channel four or by turning up the volume on the set.

Because each local device 14, by definition, has a processor with limited computing power, the respective resident VUI for a local device 14, taken alone, generally does not provide extensive speech recognition and/or speech output capability. For example, rather than implement a more complex and sophisticated natural language (NL) technique for speech recognition, each resident VUI may perform "word spotting" by scanning speech input for the occurrence of one or more "keywords." Furthermore, each local device 14 will have a relatively limited vocabulary (e.g., less than one hundred words) for its resident VUI. As such, a local device 14, by itself, is only capable of responding to relatively simple commands, instructions, directions, or requests from a user.

In instances where the speech recognition and/or speech output capability provided by a resident VUI of a local device 14 is not adequate to address the needs of a user, the resident VUI can be supplemented with the more extensive capability provided by remote system 12. Thus, the local device 14 can be controlled by spoken commands and otherwise actively participate in verbal exchanges with the user by utilizing more complex speech recognition/output hardware and/or software implemented at remote system 12 (as further described herein).

Each local device 14 may further comprise a manual input device--such as a button, a toggle switch, a keypad, or the like--by which a user can interact with the local device 14 (and also remote system 12 via a suitable communication network) to input commands, instructions, requests, or directions without using either the resident or distributed VUI. For example, each local device 14 may include hardware and/or software supporting the interpretation and issuance of dual tone multiple frequency (DTMF) commands. In one embodiment, such manual input device can be used by the user to activate or turn on the respective local device 14 and/or initiate communication with remote system 12.

Remote System

In general, remote system 12 supports a relatively sophisticated VUI which can be utilized when the capabilities of any given local device 14 alone are insufficient to address or respond to instructions, commands, directions, or requests

issued by a user at the local device 14. The VUI at remote system 12 can be implemented with speech recognition/output hardware and/or software suitable for performing the functionality described herein.

The VUI of remote system 12 interprets the vocalized expressions of a user--communicated from a local device 14--so that remote system 12 may itself respond, or alternatively, direct the local device 14 to respond, to the commands, directions, instructions, requests, and other input spoken by the user. As such, remote system 12 completes the task of recognizing words and phrases.

The VUI at remote system 12 can be implemented with a different type of automatic speech recognition (ASR) hardware/software than local devices 14. For example, in one embodiment, rather than performing "word spotting," as may occur at local devices 14, remote system 12 may use a larger vocabulary recognizer, implemented with word and optional sentence recognition grammars. A recognition grammar specifies a set of directions, commands, instructions, or requests that, when spoken by a user, can be understood by a VUI. In other words, a recognition grammar specifies what sentences and phrases are to be recognized by the VUI. For example, if a local device 14 comprises a microwave oven, a distributed VUI for the same can include a recognition grammar that allows a user to set a cooking time by saying, "Oven high for half a minute," or "Cook on high for thirty seconds," or, alternatively, "Please cook for thirty seconds at high." Commercially available speech recognition systems with recognition grammars are provided by ASR technology vendors such as, for example, the following: Nuance Corporation of Menlo Park, CA; Dragon Systems of Newton, MA; IBM of Austin, TX; Kurzweil Applied Intelligence of Waltham, MA; Lernout Hauspie Speech Products of Burlington, MA; and PureSpeech, Inc. of Cambridge, MA.

Remote system 12 may process the directions, commands, instructions, or requests that it has recognized or understood from the utterances of a user. During processing, remote system 12 can, among other things, generate control signals and reply messages, which are returned to a local device 14. Control signals are used to direct or control the local device 14 in response to user input. For example, in response to a user command of "Turn up the heat to 82 degrees," control signals may direct a local device 14 incorporating a thermostat to adjust the temperature of a climate control system. Reply messages are intended for the immediate consumption

of a user at the local device and may take the form of video or audio, or text to be displayed at the local device. As a reply message, the VUI at remote system 12 may issue audible output in the form of speech that is understandable by a user.

For issuing reply messages, the VUI of remote system 12 may include capability for speech generation (synthesized speech) and/or play-back (previously recorded speech). Speech generation capability can be implemented with text-to-speech (TTS) hardware/ software, which converts textual information into synthesized, audible speech. Speech play-back capability may be implemented with an analog-to-digital (A/D) converter driven by CD ROM (or other digital memory device), a tape player, a laser disc player, a specialized integrated circuit (IC) device, or the like, which plays back previously recorded human speech.

In speech play-back, a person (preferably a voice model) recites various statements which may desirably be issued during an interactive session with a user at a local device 14 of distributed VUI system 10. The person's voice is recorded as the recitations are made. The recordings are separated into discrete messages, each message comprising one or more statements that would desirably be issued in a particular context (e.g., greeting, farewell, requesting instructions, receiving instructions, etc.). Afterwards, when a user interacts with distributed VUI system 10, the recorded messages are played back to the user when the proper context arises.

The reply messages generated by the VUI at remote system 12 can be made to be consistent with any messages provided by the resident VUI of a local device 14. For example, if speech play-back capability is used for generating speech, the same person's voice may be recorded for messages output by the resident VUI of the local device 14 and the VUI of remote system 12. If synthesized (computer-generated) speech capability is used, a similar sounding artificial voice may be provided for the VUIs of both local devices 14 and remote system 12. In this way, the distributed VUI of system 10 provides to a user an interactive interface which is "seamless" in the sense that the user cannot distinguish between the simpler, resident VUI of the local device 14 and the more sophisticated VUI of remote system 12.

In one embodiment, the speech recognition and speech play-back capabilities described herein can be used to implement a voice user interface with personality, as taught by United States Patent Application Serial No. 09/071,717, entitled "Voice

User Interface With Personality," the text of which is incorporated herein by reference.

Remote system 12 may also comprise hardware and/or software supporting the interpretation and issuance of commands, such as dual tone multiple frequency (DTMF) commands, so that a user may alternatively interact with remote system 12 using an alternative input device, such as a telephone key pad.

Remote system 12 may be in communication with the "Internet," thus providing access thereto for users at local devices 14. The Internet is an interconnection of computer "clients" and "servers" located throughout the world and exchanging information according to Transmission Control Protocol/Internet Protocol (TCP/IP), Internetwork Packet eXchange/Sequence Packet eXchange (IPX/SPX), AppleTalk, or other suitable protocol. The Internet supports the distributed application known as the "World Wide Web." Web servers may exchange information with one another using a protocol known as hypertext transport protocol (HTTP). Information may be communicated from one server to any other computer using HTTP and is maintained in the form of web pages, each of which can be identified by a respective uniform resource locator (URL). Remote system 12 may function as a client to interconnect with Web servers. The interconnection may use any of a variety of communication links, such as, for example, a local telephone communication line or a dedicated communication line. Remote system 12 may comprise and locally execute a "web browser" or "web proxy" program. A web browser is a computer program that allows remote system 12, acting as a client, to exchange information with the World Wide Web. Any of a variety of web browsers are available, such as NETSCAPE NAVIGATOR from Netscape Communications Corp. of Mountain View, CA, INTERNET EXPLORER from Microsoft Corporation of Redmond, WA, and others that allow users to conveniently access and navigate the Internet. A web proxy is a computer program which (via the Internet) can, for example, electronically integrate the systems of a company and its vendors and/or customers, support business transacted electronically over the network (i.e., "e-commerce"), and provide automated access to Web-enabled resources. Any number of web proxies are available, such as B2B INTEGRATION SERVER from webMethods of Fairfax, VA, and MICROSOFT PROXY SERVER from Microsoft Corporation of Redmond, WA. The hardware, software, and protocols--as well as the

underlying concepts and techniques--supporting the Internet are generally understood by those in the art.

Communication Network

One or more suitable communication networks enable local devices 14 to communicate with remote system 12. For example, as shown, local devices 14a, 14b, and 14c communicate with remote system 12 via telecommunications network 16; local devices 14d, 14e, and 14f communicate via local area network (LAN) 18; and local devices 14g, 14h, and 14i communicate via the Internet.

Telecommunications network 16 allows a user to interact with remote system 12 from a local device 14 via a telecommunications line, such as an analog telephone line, a digital T1 line, a digital T3 line, or an OC3 telephony feed. Telecommunications network 16 may include a public switched telephone network (PSTN) and/or a private system (e.g., cellular system) implemented with a number of switches, wire lines, fiber-optic cable, land-based transmission towers, space-based satellite transponders, etc. In one embodiment, telecommunications network 16 may include any other suitable communication system, such as a specialized mobile radio (SMR) system. As such, telecommunications network 16 may support a variety of communications, including, but not limited to, local telephony, toll (i.e., long distance), and wireless (e.g., analog cellular system, digital cellular system, Personal Communication System (PCS), Cellular Digital Packet Data (CDPD), ARDIS, RAM Mobile Data, Metricom Ricochet, paging, and Enhanced Specialized Mobile Radio (ESMR)). Telecommunications network 16 may utilize various calling protocols (e.g., Inband, Integrated Services Digital Network (ISDN) and Signaling System No. 7 (SS7) call protocols) and other suitable protocols (e.g., Enhanced Throughput Cellular (ETC), Enhanced Cellular Control (EC^2), MNP10, MNP10-EC, Throughput Accelerator (TXCEL), Mobile Data Link Protocol, etc.). Transmissions over telecommunications network system 16 may be analog or digital. Transmission may also include one or more infrared links (e.g., IRDA).

In general, local area network (LAN) 18 connects a number of hardware devices in one or more of various configurations or topologies, which may include, for example, Ethernet, token ring, and star, and provides a path (e.g., bus) which allows the devices to communicate with each other. With local area network 18,

multiple users are given access to a central resource. As depicted, users at local devices 14d, 14e, and 14f are given access to remote system 12 for provision of the distributed VUI.

For communication over the Internet, remote system 12 and/or local devices 14g, 14h, and 14i may be connected to, or incorporate, servers and clients communicating with each other using the protocols (e.g., TCP/IP or UDP), addresses (e.g., URL), links (e.g., dedicated line), and browsers (e.g., NETSCAPE NAVIGATOR) described above.

As an alternative, or in addition, to telecommunications network 16, local area network 18, or the Internet (as depicted in FIGURE 17), distributed VUI system 10 may utilize one or more other suitable communication networks. Such other communication networks may comprise any suitable technologies for transmitting/receiving analog or digital signals. For example, such communication networks may comprise cable modems, satellite, radio, and/or infrared links.

The connection provided by any suitable communication network (e.g., telecommunications network 16, local area network 18, or the Internet) can be transient. That is, the communication network need not continuously support communication between local devices 14 and remote system 12, but rather, only provides data and signal transfer therebetween when a local device 14 requires assistance from remote system 12. Accordingly, operating costs (e.g., telephone facility charges) for distributed VUI system 10 can be substantially reduced or minimized.

Operation of Voice Gateway (In General)

In generalized operation, each local device 14 can receive input in the form of vocalized expressions (i.e., speech input) from a user and may perform preliminary or initial signal processing, such as, for example, feature extraction computations and elementary speech recognition computations. The local device 14 then determines whether it is capable of further responding to the speech input from the user. If not, local device 14 communicates--for example, over a suitable network, such as telecommunications network 16 or local area network (LAN) 18--with remote system 12. Remote system 12 performs its own processing, which may include more advanced speech recognition techniques and the accessing of other resources (e.g.,

data available on the Internet). Afterwards, remote system 12 returns a response to the local device 14. Such response can be in the form of one or more reply messages and/or control signals. The local device 14 delivers the messages to its user, and the control signals modify the operation of the local device 14.

Local Device (Details)

FIGURE 18 illustrates details for a local device 14, according to an embodiment of the present invention. As depicted, local device 14 comprises a primary functionality component 19, a microphone 20, a speaker 22, a manual input device 24, a display 26, a processing component 28, a recording device 30, and a transceiver 32.

Primary functionality component 19 performs the primary functions for which the respective local device 14 is provided. For example, if local device 14 comprises a personal digital assistant (PDA), primary functionality component 19 can maintain a personal organizer which stores information for names, addresses, telephone numbers, important dates, appointments, and the like. Similarly, if local device 14 comprises a stereo system, primary functionality component 19 can output audible sounds for a user's enjoyment by tuning into radio stations, playing tapes or compact discs, etc. If local device 14 comprises a microwave oven, primary functionality component 19 can cook foods. Primary functionality component 19 may be controlled by control signals which are generated by the remainder of local device 14, or remote system 12, in response to a user's commands, instructions, directions, or requests. Primary functionality component 19 is optional, and therefore, may not be present in every implementation of a local device 14; such a device could be one having a sole purpose of sending or transmitting information.

Microphone 20 detects the audible expressions issued by a user and relays the same to processing component 28 for processing within a parameter extraction component 34 and/or a resident voice user interface (VUI) 36 contained therein. Speaker 22 outputs audible messages or prompts which can originate from resident VUI 36 of local device 14, or alternatively, from the VUI at remote system 12. Speaker 22 is optional, and therefore, may not be present in every implementation; for example, a local device 14 can be implemented such that output to a user is via display 26 or primary functionality component 19.

Manual input device 24 comprises a device by which a user can manually input information into local device 14 for any of a variety of purposes. For example, manual input device 24 may comprise a keypad, button, switch, or the like, which a user can depress or move to activate/deactivate local device 14, control local device 14, initiate communication with remote system 12, input data to remote system 12, etc. Manual input device 24 is optional, and therefore, may not be present in every implementation; for example, a local device 14 can be implemented such that user input is via microphone 20 only. Display 26 comprises a device, such as, for example, a liquid-crystal display (LCD) or light-emitting diode (LED) screen, which displays data visually to a user. In some embodiments, display 26 may comprise an interface to another device, such as a television set. Display 26 is optional, and therefore, may not be present in every implementation; for example, a local device 14 can be implemented such that user output is via speaker 22 only.

Processing component 28 is connected to each of primary functionality component 19, microphone 20, speaker 22, manual input device 24, and display 26. In general, processing component 28 provides processing or computing capability in local device 14. In one embodiment, processing component 28 may comprise a microprocessor connected to (or incorporating) supporting memory to provide the functionality described herein. As previously discussed, such a processor has limited computing power.

Processing component 28 may output control signals to primary functionality component 19 for control thereof. Such control signals can be generated in response to commands, instructions, directions, or requests which are spoken by a user and interpreted or recognized by resident VUI 36 and/or remote system 12. For example, if local device 14 comprises a household security system, processing component 28 may output control signals for disarming the security system in response to a user's verbalized command of "Security off, code 4-2-5-6-7."

Parameter extraction component 34 may perform a number of preliminary signal processing operations on a speech waveform. Among other things, these operations transform speech into a series of feature parameters, such as standard cepstral coefficients, Fourier coefficients, linear predictive coding (LPC) coefficients, or other parameters in the frequency or time domain. For example, in one embodiment, parameter extraction component 34 may produce a twelve-dimensional

vector of cepstral coefficients every ten milliseconds to model speech input data. Software for implementing parameter extraction component 34 is commercially available from line card manufacturers and ASR technology suppliers such as Dialogic Corporation of Parsippany, NJ, and Natural MicroSystems Inc. of Natick, MA.

Resident VUI 36 may be implemented in processing component 28. In general, VUI 36 allows local device 14 to understand and speak to a user on at least an elementary level. As shown, VUI 36 of local device 14 may include a barge-in component 38, a speech recognition engine 40, and a speech generation engine 42.

Barge-in component 38 generally functions to detect speech from a user at microphone 20 and, in one embodiment, can distinguish human speech from ambient background noise. When speech is detected by barge-in component 38, processing component 28 ceases to emit any speech which it may currently be outputting so that processing component 28 can attend to the new speech input. Thus, a user is given the impression that he or she can interrupt the speech generated by local device 14 (and the distributed VUI system 10) simply by talking. Software for implementing barge-in component 38 is commercially available from line card manufacturers and ASR technology suppliers such as Dialogic Corporation of Parsippany, NJ, and Natural MicroSystems Inc. of Natick, MA. Barge-in component 38 is optional, and therefore, may not be present in every implementation.

Speech recognition engine 40 can recognize speech at an elementary level, for example, by performing keyword searching. For this purpose, speech recognition engine 40 may comprise a keyword search component 44 which is able to identify and recognize a limited number (e.g., 100 or less) of keywords. Each keyword may be selected in advance based upon commands, instructions, directions, or requests which are expected to be issued by a user. In one embodiment, speech recognition engine 40 may comprise a logic state machine. Speech recognition engine 40 can be implemented with automatic speech recognition (ASR) software commercially available, for example, from the following companies: Nuance Corporation of Menlo Park, CA; Applied Language Technologies, Inc. of Boston, MA; Dragon Systems of Newton, MA; and PureSpeech, Inc. of Cambridge, MA. Such commercially available software typically can be modified for particular applications, such as a computer telephony application. As such, the resident VUI 36 can be configured or modified by

a user or another party to include a customized keyword grammar. In one embodiment, keywords for a grammar can be downloaded from remote system 12. In this way, keywords already existing in local device 14 can be replaced, supplemented, or updated as desired.

Speech generation engine 42 can output speech, for example, by playing back pre-recorded messages, to a user at appropriate times. For example, several recorded prompts and/or responses can be stored in the memory of processing component 28 and played back at any appropriate time. Such play-back capability can be implemented with a play-back component 46 comprising suitable hardware/software, which may include an integrated circuit device. In one embodiment, pre-recorded messages (e.g., prompts and responses) may be downloaded from remote system 12. In this manner, the pre-recorded messages already existing in local device 14 can be replaced, supplemented, or updated as desired. Speech generation engine 42 is optional, and therefore, may not be present in every implementation; for example, a local device 14 can be implemented such that user output is via display 26 or primary functionality component 19 only.

Recording device 30, which is connected to processing component 28, functions to maintain a record of each interactive session with a user (i.e., interaction between distributed VUI system 10 and a user after activation, as described below). Such record may include the verbal utterances issued by a user during a session and preliminarily processed by parameter extraction component 34 and/or resident VUI 36. These recorded utterances are exemplary of the language used by a user and also the acoustic properties of the user's voice. The recorded utterances can be forwarded to remote system 12 for further processing and/or recognition. In a robust technique, the recorded utterances can be analyzed (for example, at remote system 12) and the keywords recognizable by distributed VUI system 10 updated or modified according to the user's word choices. The record maintained at recording device 30 may also specify details for the resources or components used in maintaining, supporting, or processing the interactive session. Such resources or components can include microphone 20, speaker 22, telecommunications network 16, local area network 18, connection charges (e.g., telecommunications charges), etc. Recording device 30 can be implemented with any suitable hardware/software. Recording device 30 is optional, and therefore, may not be present in some implementations.

Transceiver 32 is connected to processing component 28 and functions to provide bi-directional communication with remote system 12 over telecommunications network 16. Among other things, transceiver 32 may transfer speech and other data to and from local device 14. Such data may be coded, for example, using 32-KB Adaptive Differential Pulse Coded Modulation (ADPCM) or 64-KB MU-law parameters using commercially available modulation devices from, for example, Rockwell International of Newport Beach, CA. In addition, or alternatively, speech data may be transfer coded as LPC parameters or other parameters achieving low bit rates (e.g., 4.8 Kbits/sec), or using a compressed format, such as, for example, with commercially available software from Voxware of Princeton, New Jersey. Data sent to remote system 12 can include frequency domain parameters extracted from speech by processing component 28. Data received from remote system 12 can include that supporting audio and/or video output at local device 14, and also control signals for controlling primary functionality component 19. The connection for transmitting data to remote system 12 can be the same or different from the connection for receiving data from remote system 12. In one embodiment, a "high bandwidth" connection is used to return data for supporting audio and/or video, whereas a "low bandwidth" connection may be used to return control signals.

In one embodiment, in addition to, or in lieu of, transceiver 32, local device 14 may comprise a local area network (LAN) connector and/or a wide area network (WAN) connector (neither of which are explicitly shown) for communicating with remote system 12 via local area network 18 or the Internet, respectively. The LAN connector can be implemented with any device which is suitable for the configuration or topology (e.g., Ethernet, token ring, or star) of local area network 18. The WAN connector can be implemented with any device (e.g., router) supporting an applicable protocol (e.g., TCP/IP, IPX/SPX, or AppleTalk).

Local device 14 may be activated upon the occurrence of any one or more activation or triggering events. For example, local device 14 may activate at a predetermined time (e.g., 7:00 a.m. each day), at the lapse of a predetermined interval (e.g., twenty-four hours), or upon triggering by a user at manual input device 24. Alternatively, resident VUI 36 of local device 14 may be constantly operating--

listening to speech issued from a user, extracting feature parameters (e.g., cepstral, Fourier, or LPC) from the speech, and/or scanning for keyword "wake up" phrases.

After activation and during operation, when a user verbally issues commands, instructions, directions, or requests at microphone 20 or inputs the same at manual input device 24, local device 14 may respond by outputting control signals to primary functionality component 19 and/or outputting speech to the user at speaker 22. If local device 14 is able, it generates these control signals and/or speech by itself after processing the user's commands, instructions, directions, or requests, for example, within resident VUI 36. If local device 14 is not able to respond by itself (e.g., it cannot recognize a user's spoken command) or, alternatively, if a user triggers local device 14 with a "wake up" command, local device 14 initiates communication with remote system 12. Remote system 12 may then process the spoken commands, instructions, directions, or requests at its own VUI and return control signals or speech to local device 14 for forwarding to primary functionality component 19 or a user, respectively.

For example, local device 14 may, by itself, be able to recognize and respond to an instruction of "Dial number 555-1212," but may require the assistance of remote device 12 to respond to a request of "What is the weather like in Chicago?"

Remote System (Details)

FIGURE 19 illustrates details for a remote system 12, according to an embodiment of the present invention. Remote system 12 may cooperate with local devices 14 to provide a distributed VUI for communication with respective users and to generate control signals for controlling respective primary functionality components 19. As depicted, remote system 12 comprises a transceiver 50, a LAN connector 52, a processing component 54, a memory 56, and a WAN connector 58. Depending on the combination of local devices 14 supported by remote system 12, only one of the following may be required, with the other two optional: transceiver 50, LAN connector 52, or WAN connector 58.

Transceiver 50 provides bi-directional communication with one or more local devices 14 over telecommunications network 16. As shown, transceiver 50 may include a telephone line card 60 which allows remote system 12 to communicate with telephone lines, such as, for example, analog telephone lines, digital T1 lines, digital

T3 lines, or OC3 telephony feeds. Telephone line card 60 can be implemented with various commercially available telephone line cards from, for example, Dialogic Corporation of Parsippany, NJ (which supports twenty-four lines) or Natural MicroSystems Inc. of Natick, MA (which supports from two to forty-eight lines). Among other things, transceiver 50 may transfer speech data to and from local device 14. Speech data can be coded as, for example, 32-KB Adaptive Differential Pulse Coded Modulation (ADPCM) or 64-KB MU-law parameters using commercially available modulation devices from, for example, Rockwell International of Newport Beach, CA. In addition, or alternatively, speech data may be transfer coded as LPC parameters or other parameters achieving low bit rates (e.g., 4.8 Kbits/sec), or using a compressed format, such as, for example, with commercially available software from Voxware of Princeton, New Jersey.

LAN connector 52 allows remote system 12 to communicate with one or more local devices over local area network 18. LAN connector 52 can be implemented with any device supporting the configuration or topology (e.g., Ethernet, token ring, or star) of local area network 18. LAN connector 52 can be implemented with a LAN card commercially available from, for example, 3COM Corporation of Santa Clara, California.

Processing component 54 is connected to transceiver 50 and LAN connector 52. In general, processing component 54 provides processing or computing capability in remote system 12. The functionality of processing component 54 can be performed by any suitable processor, such as a main-frame, a file server, a workstation, or other suitable data processing facility supported by memory (either internal or external) and running appropriate software. In one embodiment, processing component 54 can be implemented as a physically distributed or replicated system. Processing component 54 may operate under the control of any suitable operating system (OS), such as MS-DOS, MacINTOSH OS, WINDOWS NT, WINDOWS 95, OS/2, UNIX, LINUX, XENIX, and the like.

Processing component 54 may receive--from transceiver 50, LAN connector 52, and WAN connector 58 --commands, instructions, directions, or requests, issued by one or more users at local devices 14. Processing component 54 processes these user commands, instructions, directions, or requests and, in response, may generate control signals or speech output.

For recognizing and outputting speech, a VUI 62 is implemented in processing component 54. This VUI 62 is more sophisticated than the resident VUIs 34 of local devices 14. For example, VUI 62 can have a more extensive vocabulary with respect to both the word/phrases which are recognized and those which are output. VUI 62 of remote system 12 can be made to be consistent with resident VUIs 34 of local devices 14. For example, the messages or prompts output by VUI 62 and VUIs 34 can be generated in the same synthesized, artificial voice. Thus, VUI 62 and VUIs 34 operate to deliver a "seamless" interactive interface to a user. In some embodiments, multiple instances of VUI 62 may be provided such that a different VUI is used based on the type of local device 14. As shown, VUI 62 of remote system 12 may include an echo cancellation component 64, a barge-in component 66, a signal processing component 68, a speech recognition engine 70, and a speech generation engine 72.

Echo cancellation component 64 removes echoes caused by delays (e.g., in telecommunications network 16) or reflections from acoustic waves in the immediate environment of a local device 14. This provides "higher quality" speech for recognition and processing by VUI 62. Software for implementing echo cancellation component 64 is commercially available from Noise Cancellation Technologies of Stamford, CN.

Barge-in component 66 may detect speech received at transceiver 50, LAN connector 52, or WAN connector 58. In one embodiment, barge-in component 66 may distinguish human speech from ambient background noise. When barge-in component 66 detects speech, any speech output by the distributed VUI is halted so that VUI 62 can attend to the new speech input. Software for implementing barge-in component 66 is commercially available from line card manufacturers and ASR technology suppliers such as, for example, Dialogic Corporation of Parsippany, NJ, and Natural MicroSystems Inc. of Natick, MA. Barge-in component 66 is optional, and therefore, may not be present in every implementation.

Signal processing component 68 performs signal processing operations which, among other things, may include transforming speech data received in time domain format (such as ADPCM) into a series of feature parameters such as, for example, standard cepstral coefficients, Fourier coefficients, linear predictive coding (LPC) coefficients, or other parameters in the time or frequency domain. For example, in one embodiment, signal processing component 68 may produce a twelve-dimensional

vector of cepstral coefficients every 10 milliseconds to model speech input data. Software for implementing signal processing component 68 is commercially available from line card manufacturers and ASR technology suppliers such as Dialogic Corporation of Parsippany, NJ, and Natural MicroSystems Inc. of Natick, MA.

Speech recognition engine 70 allows remote system 12 to recognize vocalized speech. As shown, speech recognition engine 70 may comprise an acoustic model component 73 and a grammar component 74. Acoustic model component 73 may comprise one or more reference voice templates which store previous enunciations (or acoustic models) of certain words or phrases by particular users. Acoustic model component 73 recognizes the speech of the same users based upon their previous enunciations stored in the reference voice templates. Grammar component 74 may specify certain words, phrases, and/or sentences which are to be recognized if spoken by a user. Recognition grammars for grammar component 74 can be defined in a grammar definition language (GDL), and the recognition grammars specified in GDL can then be automatically translated into machine executable grammars. In one embodiment, grammar component 74 may also perform natural language (NL) processing. Hardware and/or software for implementing a recognition grammar is commercially available from such vendors as the following: Nuance Corporation of Menlo Park, CA; Dragon Systems of Newton, MA; IBM of Austin, TX; Kurzweil Applied Intelligence of Waltham, MA; Lernout Hauspie Speech Products of Burlington, MA; and PureSpeech, Inc. of Cambridge, MA. Natural language processing techniques can be implemented with commercial software products separately available from, for example, UNISYS Corporation of Blue Bell, PA. These commercially available hardware/software can typically be modified for particular applications.

Speech generation engine 72 allows remote system 12 to issue verbalized responses, prompts, or other messages, which are intended to be heard by a user at a local device 14. As depicted, speech generation engine 72 comprises a text-to-speech (TTS) component 76 and a play-back component 78. Text-to-speech component 76 synthesizes human speech by "speaking" text, such as that contained in a textual e-mail document. Text-to-speech component 76 may utilize one or more synthetic speech mark-up files for determining, or containing, the speech to be synthesized. Software for implementing text-to-speech component 76 is commercially available,

for example, from the following companies: AcuVoice, Inc. of San Jose, CA; Centigram Communications Corporation of San Jose, CA; Digital Equipment Corporation (DEC) of Maynard, MA; Lucent Technologies of Murray Hill, NJ; and Entropic Research Laboratory, Inc. of Washington, D.C. Play-back component 78 plays back pre-recorded messages to a user. For example, several thousand recorded prompts or responses can be stored in memory 56 of remote system 12 and played back at any appropriate time. Speech generation engine 72 is optional (including either or both of text-to-speech component 76 and play-back component 78), and therefore, may not be present in every implementation.

Memory 56 is connected to processing component 54. Memory 56 may comprise any suitable storage medium or media, such as random access memory (RAM), read-only memory (ROM), disk, tape storage, or other suitable volatile and/or non-volatile data storage system. Memory 56 may comprise a relational database. Memory 56 receives, stores, and forwards information which is utilized within remote system 12 and, more generally, within distributed VUI system 10. For example, memory 56 may store the software code and data supporting the acoustic models, grammars, text-to-speech, and play-back capabilities of speech recognition engine 70 and speech generation engine 72 within VUI 64.

WAN connector 58 is coupled to processing component 54. WAN connector 58 enables remote system 12 to communicate with the Internet using, for example, Transmission Control Protocol/Internet Protocol (TCP/IP), Internetwork Packet eXchange/Sequence Packet eXchange (IPX/SPX), AppleTalk, or any other suitable protocol. By supporting communication with the Internet, WAN connector 58 allows remote system 12 to access various remote databases containing a wealth of information (e.g., stock quotes, telephone listings, directions, news reports, weather and travel information, etc.) which can be retrieved/downloaded and ultimately relayed to a user at a local device 14. WAN connector 58 can be implemented with any suitable device or combination of devices--such as, for example, one or more routers and/or switches--operating in conjunction with suitable software. In one embodiment, WAN connector 58 supports communication between remote system 12 and one or more local devices 14 over the Internet.

Operation at Local Device

FIGURE 20 is a flow diagram of an exemplary method 100 of operation for a local device 14, according to an embodiment of the present invention.

Method 100 begins at step 102 where local device 14 waits for some activation event, or particular speech issued from a user, which initiates an interactive user session, thereby activating processing within local device 14. Such activation event may comprise the lapse of a predetermined interval (e.g., twenty-four hours) or triggering by a user at manual input device 24, or may coincide with a predetermined time (e.g., 7:00 a.m. each day). In another embodiment, the activation event can be speech from a user. Such speech may comprise one or more commands in the form of keywords--e.g., "Start," "Turn on," or simply "On"--which are recognizable by resident VUI 36 of local device 14. If nothing has occurred to activate or start processing within local device 14, method 100 repeats step 102. When an activating event does occur, and hence, processing is initiated within local device 14, method 100 moves to step 104.

At step 104, local device 14 receives speech input from a user at microphone 20. This speech input--which may comprise audible expressions of commands, instructions, directions, or requests spoken by the user--is forwarded to processing component 28. At step 106 processing component 28 processes the speech input. Such processing may comprise preliminary signal processing, which can include parameter extraction and/or speech recognition. For parameter extraction, parameter extraction component 34 transforms the speech input into a series of feature parameters, such as standard cepstral coefficients, Fourier coefficients, LPC coefficients, or other parameters in the time or frequency domain. For speech recognition, resident VUI 36 distinguishes speech using barge-in component 38, and may recognize speech at an elementary level (e.g., by performing key-word searching), using speech recognition engine 40.

As speech input is processed, processing component 28 may generate one or more responses. Such response can be a verbalized response which is generated by speech generation engine 42 and output to a user at speaker 22. Alternatively, the response can be in the form of one or more control signals, which are output from processing component 28 to primary functionality component 19 for control thereof. Steps 104 and 106 may be repeated multiple times for various speech input received from a user.

At step 108, processing component 28 determines whether processing of speech input locally at local device 14 is sufficient to address the commands, instructions, directions, or requests from a user. If so, method 100 proceeds to step 120 where local device 14 takes action based on the processing, for example, by replying to a user and/or controlling primary functionality component 19. Otherwise, if local processing is not sufficient, then at step 110, local device 14 establishes a connection between itself and remote device 12, for example, via telecommunications network 16 or local area network 18.

At step 112, local device 14 transmits data and/or speech input to remote system 12 for processing therein. Local device 14 at step 113 then waits, for a predetermined period, for a reply or response from remote system 12. At step 114, local device 14 determines whether a time-out has occurred--i.e., whether remote system 12 has failed to reply within a predetermined amount of time allotted for response. A response from remote system 12 may comprise data for producing an audio and/or video output to a user, and/or control signals for controlling local device 14 (especially, primary functionality component 19).

If it is determined at step 114 that remote system 12 has not replied within the time-out period, local device 14 may terminate processing, and method 100 ends. Otherwise, if a time-out has not yet occurred, then at step 116 processing component 28 determines whether a response has been received from remote system 12. If no response has yet been received from remote system 12, method 100 returns to step 113 where local device 14 continues to wait. Local device 14 repeats steps 113, 114, and 116 until either the time-out period has lapsed or, alternatively, a response has been received from remote system 12.

After a response has been received from remote system 12, then at step 118 local device 14 may terminate the connection between itself and remote device 12. In one embodiment, if the connection comprises a toll-bearing public switched telephone network (PSTN) connection, termination can be automatic (e.g., after the lapse of a time-out period). In another embodiment, termination is user-activated; for example, the user may enter a predetermined series of dual tone multiple frequency (DTMF) signals at manual input device 24.

At step 120, local device 14 takes action based upon the response from remote system 12. This may include outputting a reply message (audible or visible) to the user and/or controlling the operation of primary functionality component 19.

At step 122, local device 14 determines whether this interactive session with a user should be ended. For example, in one embodiment, a user may indicate his or her desire to end the session by ceasing to interact with local device 14 for a predetermined (time-out) period, or by entering a predetermined series of dual tone multiple frequency (DTMF) signals at manual input device 24. If it is determined at step 122 that the interactive session should not be ended, then method 100 returns to step 104 where local device 14 receives speech from a user. Otherwise, if it is determined that the session should be ended, method 100 ends.

Operation at Remote System

FIGURE 21 is a flow diagram of an exemplary method 200 of operation for remote system 12, according to an embodiment of the present invention.

Method 200 begins at step 202 where remote system 12 awaits user input from a local device 14. Such input--which may be received at transceiver 50, LAN connector 52, or WAN connector 58--may specify a command, instruction, direction, or request from a user. The input can be in the form of data, such as a DTMF signal or speech. When remote system 12 has received an input, such input is forwarded to processing component 54.

Processing component 54 then processes or operates upon the received input. For example, assuming that the input is in the form of speech, echo cancellation component 64 of VUI 62 may remove echoes caused by transmission delays or reflections, and barge-in component 66 may detect the onset of human speech. Furthermore, at step 204, speech recognition engine 70 of VUI 62 compares the command, instruction, direction, or request specified in the input against grammars which are contained in grammar component 74. These grammars may specify certain words, phrases, and/or sentences which are to be recognized if spoken by a user. Alternatively, speech recognition engine 70 may compare the speech input against one or more acoustic models contained in acoustic model component 73.

At step 206, processing component 62 determines whether there is a match between the verbalized command, instruction, direction, or request spoken by a user

and a grammar (or acoustic model) recognizable by speech recognition engine 70. If so, method 200 proceeds to step 224 where remote system 12 responds to the recognized command, instruction, direction, or request, as further described below. On the other hand, if it is determined at step 206 that there is no match (between a grammar (or acoustic model) and the user's spoken command, instruction, direction, or request), then at step 208 remote system 12 requests more input from a user. This can be accomplished, for example, by generating a spoken request in speech generation engine 72 (using either text-to-speech component 76 or play-back component 78) and then forwarding such request to local device 14 for output to the user.

When remote system 12 has received more spoken input from the user (at transceiver 50, LAN connector 52, or WAN connector 58), processing component 54 again processes the received input (for example, using echo cancellation component 64 and barge-in component 66). At step 210, speech recognition engine 70 compares the most recently received speech input against the grammars of grammar component 74 (or the acoustic models of acoustic model component 73).

At step 212, processing component 54 determines whether there is a match between the additional input and the grammars (or the acoustic models). If there is a match, method 200 proceeds to step 224. Alternatively, if there is no match, then at step 214 processing component 54 determines whether remote system 12 should again attempt to solicit speech input from the user. In one embodiment, a predetermined number of attempts may be provided for a user to input speech; a counter for keeping track of these attempts is reset each time method 200 performs step 202, where input speech is initially received. If it is determined that there are additional attempts left, then method 200 returns to step 208 where remote system 12 requests (via local device 14) more input from a user.

Otherwise, method 200 moves to step 216 where processing component 54 generates a message directing the user to select from a list of commands or requests which are recognizable by VUI 62. This message is forwarded to local device 14 for output to the user. For example, in one embodiment, the list of commands or requests is displayed to a user on display 26. Alternatively, the list can be spoken to the user via speaker 22.

In response to the message, the user may then select from the list by speaking one or more of the commands or requests. This speech input is then forwarded to remote system 12. At step 218, speech recognition engine 70 of VUI 62 compares the speech input against the grammars (or the acoustic models) contained therein.

At step 220, processing component 54 determines whether there is a match between the additional input and the grammars (or the acoustic models). If there is a match, method 200 proceeds to step 224. Otherwise, if there is no match, then at step 222 processing component 54 determines whether remote system 12 should again attempt to solicit speech input from the user by having the user select from the list of recognizable commands or requests. In one embodiment, a predetermined number of attempts may be provided for a user to input speech in this way; a counter for keeping track of these attempts is reset each time method 200 performs step 202, where input speech is initially received. If it is determined that there are additional attempts left, then method 200 returns to step 216 where remote system 12 (via local device 14) requests that the user select from the list. Alternatively, if it is determined that no attempts are left (and hence, remote system 12 has failed to receive any speech input that it can recognize), method 200 moves to step 226.

At step 224, remote system 12 responds to the command, instruction, direction or request from a user. Such response may include accessing the Internet via LAN connector 58 to retrieve requested data or information. Furthermore, such response may include generating one or more vocalized replies (for output to a user) or control signals (for directing or controlling local device 14).

At step 226, remote system 12 determines whether this session with local device 14 should be ended (for example, if a time-out period has lapsed). If not, method 200 returns to step 202 where remote system 12 waits for another command, instruction, direction, or request from a user. Otherwise, if it is determined at step 216 that there should be an end to this session, method 200 ends.

In an alternative operation, rather than passively waiting for user input from a local device 14 to initiate a session between remote system 12 and the local device, remote system 12 actively triggers such a session. For example, in one embodiment, remote system 12 may actively monitor stock prices on the Internet and initiate a

session with a relevant local device 14 to inform a user when the price of a particular stock rises above, or falls below, a predetermined level.

Accordingly, as described herein, the present invention provides a system and method for a distributed voice user interface (VUI) in which remote system 12 cooperates with one or more local devices 14 to deliver a sophisticated voice user interface at each of local devices 14.

Although particular embodiments of the present invention have been shown and described, it will be obvious to those skilled in the art that changes and modifications may be made without departing from the present invention in its broader aspects, and therefore, the appended claims are to encompass within their scope all such changes and modifications that fall within the true scope of the present invention.